EE276 Information Theory Lecture 5 - 01/23/2024

Lecture 5: Kraft-McMillan Inequality and Huffman Coding

Lecturer: Tsachy Weissman

In this lecture, we begin with an observation that Shannon coding is not optimal, in the sense that it does not in general achieve the lowest possible code word length. We will then show that entropy is the lower bound on the expected codeword length of any variable length uniquely decodable code. Finally, we introduce the Huffman Code, a method that produces optimal prefix codes for lossless compression.

1 Suboptimality of Shannon coding

Recall from the previous lecture that for a given source distribution $p(u)$, the Shannon code is a prefix code with length function and expected code length as follows:

$$l(u) = \left\lceil \log \frac{1}{p(u)} \right\rceil \quad \forall u \in \mathcal{U} \tag{1}$$

$$\mathbb{E}[l(U)] = \bar{l} \leq H(U) + 1 \tag{2}$$

Example: Consider a binary source $\mathcal{U}$ for which one symbol is much more likely to occur than the other:

$$\mathcal{U} = \{a, b\}, \quad p(a) = 0.99, \quad p(b) = 0.01 \tag{3}$$

We can construct the Shannon code for this source:

$$l(a) = \left\lceil \log \frac{1}{p(a)} \right\rceil = \lceil 0.0145 \rceil = 1 \tag{4}$$

$$l(b) = \left\lceil \log \frac{1}{p(b)} \right\rceil = \lceil 6.64 \rceil = 7 \tag{5}$$

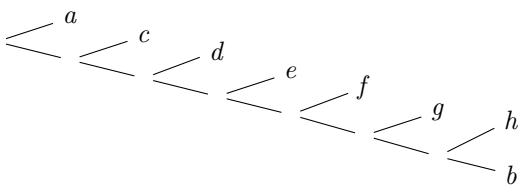
Now consider a fictitious source $p^*(u)$ with symbol probabilities as follows:

$$p^*(a) = 2^{-1} = \frac{1}{2}, \quad p^*(b) = 2^{-7} = \frac{1}{128} \tag{6}$$

And recall that we can extend this source with additional symbols to $\mathcal{U}^* \supseteq \mathcal{U}$ such that $p^*(u)$ is dyadic and $\sum_{u \in \mathcal{U}^*} p^*(u) = 1$:

$$p^*(c) = 2^{-2}, \quad p^*(d) = 2^{-3}, \quad p^*(e) = 2^{-4}, \quad p^*(f) = 2^{-5}, \quad p^*(g) = 2^{-6}, \quad p^*(h) = 2^{-7} \tag{7}$$

The binary tree representation of the code would look like this:



And the expected code length and entropy would be:

$$\bar{l} = \mathbb{E}[l(U)] = 0.99 \cdot 1 + 0.01 \cdot 7 = 1.06 \tag{8}$$

$$H(U) = h(0.01) = 0.081 \tag{9}$$

As seen here, even though we are within one bit of the entropy, for a binary source this is not a really an advantage in practice. In fact for a binary source, the optimal code would have been simply using one bit (i.e. 0) for symbol a and one bit (i.e. 1) for symbol b. We will soon learn a method to construct code for a given source such that it is optimal, i.e. achieving the lowest possible expected code length.

2 Average codelength bound for uniquely decodable codes

Theorem: (Kraft-McMillan Inequality). *For all uniquely decodable (UD) codes:*

$$\sum_{u\in\mathcal{U}} 2^{-l(u)} \leq 1 \tag{10}$$

Conversely, any integer-valued function satisfying this inequality is the length function of some UD code.

Note: The converse has already been shown to be true in earlier discussions, because if $\sum_{u\in\mathcal{U}} 2^{-l(u)} \leq 1$ holds then we can construct a Shannon *prefix* code for the dyadic source $p(u) = 2^{-l(u)}$. Thus, this theorem will imply that any set of lengths possible for a uniquely decodable code can also be achieved with prefix codes. This statement is even stronger than the statement that prefix codes can achieve the same expected length as uniquely decodable codes. This justifies our focus on the class of prefix codes.

We will now proceed with proving the first claim of the Kraft-McMillan Inequality.

Proof: Take any uniquely decodable code and let $l_{max} = \max_u l(u)$. Fix any integer k and observe:

$$\left(\sum_{u\in\mathcal{U}} 2^{-l(u)}\right)^k = \left(\sum_{u_1} 2^{-l(u_1)}\right)\cdot\left(\sum_{u_2} 2^{-l(u_2)}\right)\cdot\ldots\cdot\left(\sum_{u_k} 2^{-l(u_k)}\right) \tag{11}$$

$$= \sum_{u_1}\sum_{u_2}\ldots\sum_{u_k}\prod_{i=1}^{k} 2^{-l(u_i)} \tag{12}$$

$$= \sum_{u_1,\ldots u_k} 2^{-\sum_{i=1}^{k} l(u_i)} \tag{13}$$

$$= \sum_{u^k} 2^{-l(u^k)} \tag{14}$$

$$= \sum_{i=1}^{k\cdot l_{max}} |\{u^k | l(u^k) = i\}|\cdot 2^{-i} \tag{15}$$

Since the code is one-to-one, there is at most 2^i symbols whose codewords have length i. This is the one place where we use the assumption that we have a uniquely decodable code. So we can get an upper-bound:

$$\left(\sum_{u\in\mathcal{U}} 2^{-l(u)}\right)^k \leq \sum_{i=1}^{k\cdot l_{max}} 2^i\cdot 2^{-i} = k\cdot l_{max} \tag{16}$$

Notice that the exponential term $(\sum_{u\in\mathcal{U}} 2^{-l(u)})^k$ is in fact upper-bounded by $k\cdot l_{max}$, which is linear in k. From this we arrive at the theorem's claim (note that the inequality holds for all $k > 0$):

$$\sum_{u\in\mathcal{U}} 2^{-l(u)} \leq \lim_{k\to\infty}(k\cdot l_{max})^{1/k} = 1 \tag{17}$$

□

This leads us to an important conclusion on the length of UD codes and the entropy.

Theorem: *For all UD codes:*

$$\bar{l} \geq H(U) \tag{18}$$

Proof: Let $q(u) = c2^{-l(u)}$ be a P.M.F. where the normalizing constant c is chosen such that the probability sums to 1. By the Kraft-McMillan Inequality, we know that $c \geq 1$, since,

$$q(u) = c \cdot 2^{-l(u)}, \quad c = \frac{1}{\sum_{u \in \mathcal{U}} 2^{-l(u)}} \geq 1 \tag{19}$$

Next, consider the relative entropy between the probability distributions of the source p and q (Taking advantage of the non-negativity of relative entropy):

$$\begin{aligned}
0 &\leq D(p\|q) && (20)\\
&= \sum_u p(u) \log \frac{p(u)}{q(u)} && (21)\\
&= \sum_u p(u) \log p(u) + \sum_u p(u) \log \frac{1}{q(u)} && (22)\\
&= \sum_u p(u) \log p(u) + \sum_u p(u)[l(u) - \underbrace{\log c}_{\geq 0}] && (23)\\
&\leq -H(U) + \bar{l} && (24)
\end{aligned}$$

□

Note: Beyond its role in this proof, the relative entropy has significance in the context of compression. Suppose we have length function $l(u) \approx -\log q(u)$ for some P.M.F. q. This code will be near optimal if the source distribution is $q(u)$. However, if the true source distribution is $p(u)$, then,

$$\begin{aligned}
\bar{l} - H(U) &\approx \mathbb{E}\left[\log \frac{1}{q(u)}\right] - \mathbb{E}\left[\log \frac{1}{p(u)}\right] && (25)\\
&= \sum_u p(u) \log \frac{p(u)}{q(u)} && (26)\\
&= D(p\|q) && (27)
\end{aligned}$$

The relative entropy can be thought of as the cost of mismatch in lossless compression, i.e. the expected number of excess bits that will be expended due to optimizing the code for distribution q when the true distribution is p.

3 Huffman coding

We earlier looked at Shannon code, which is a pretty good construction of a prefix code for a given distribution. However, the best prefix code for a general source code distribution is the Huffman Code.

Our construction will be exactly like that for dyadic sources. We will recursively merge the two symbols with the smallest probabilities.

To find the code $c(u)$, we follow these steps:

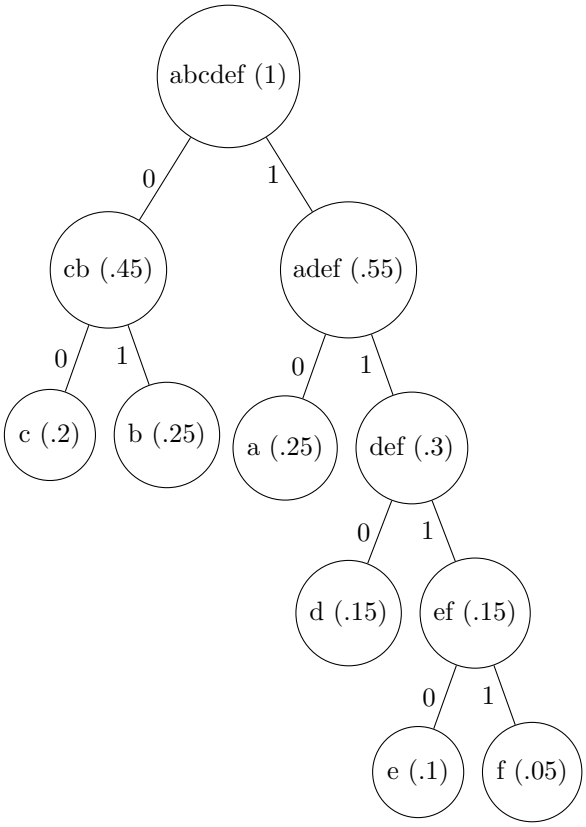
1. Find 2 symbols with the smallest probability and then merge them to create a new "node" and treat it as a new symbol.
2. Then merge the next 2 symbols with the smallest probability to create a new "node".
3. Repeat steps 1 and 2 until there is only 1 symbol left.

At the end of this process, we obtain a binary tree. The paths traversed from the root to the leaves are the prefix codes. Recall that this is a prefix code since we only assign the source symbols to the leaf nodes. Also note that the Huffman code is not necessarily unique since it depends on the order of the merge operations.

Aside: (Source arity nomenclature) *binary (2 symbols), ternary (3 symbols), quaternary (4 symbols), quinary (5 symbols), senary (6 symbols), septenary (7 symbols), octal (8 symbols), nonary (9 symbols), decimal (10 symbols) ...*

Example: Senary (6 symbol) source:

u	$p(u)$	$c(u)$	$l(u)$
a	.25	10	2
b	.25	00	2
c	.2	01	2
d	.15	110	3
e	.1	1110	4
f	.05	1111	4



The probability is included in parenthesis.

Theorem 1. *The Huffman code is an optimal prefix code.*

Proof: Assume without loss of generality that $U \sim p$, $\mathcal{U} = \{1, 2, \ldots, r\}$ and $p(1) \geq p(2) \geq \ldots \geq p(r)$. Note that we are just ordering the probabilities. If the probabilities are not ordered we can just relabel the elements so that they are ordered.

Let V denote the random variable with alphabet $\{1, 2, \ldots, r-1\}$ obtained from U by merging symbols $r-1$ and r.

Let $\{c(u)\}_{i=1}^{r-1}$ be a prefix code for V.

We can obtain a prefix code $\{\tilde{c}(i)\}_{i=1}^{r}$ for U by "splitting" the last code word.

$$\tilde{c}(i) = \begin{cases} c(i) & i = 1, 2, \ldots r-2 \\ c(r-1)0 \text{ (concatenate a 0)} & i = r-1 \\ c(r-1)1 \text{ (concatenate a 1)} & i = r \end{cases}$$

$\tilde{c}$ is a prefix code because c is a prefix code.

Note: If $\ell(\cdot)$ is the length function of $\{c(i)\}_{i=1}^{r}$ and $\ell_{split}(\cdot)$ is the length function of $\{\tilde{c}(i)\}_{i=1}^{r}$, then $\mathbb{E}[\ell_{split}(u)] = \mathbb{E}[\ell(V)] + p(r-1) + p(r)$

Proof of Note:

$$\begin{aligned} \mathbb{E}[\ell_{split}(u)] &= \sum_{u=1}^{r} p(u)\ell_{split}(u) \\ &= \sum_{u=1}^{r-2} p(u)\ell_{split}(u) + p(r-1)\ell_{split}(r-1) + p(r)\ell_{split}(r) \\ &= \sum_{u=1}^{r-2} p(u)\ell(u) + p(r-1)(\ell(r-1)+1) + p(r)(\ell(r-1)+1) \\ &= \left[\sum_{u=1}^{r-2} p(u)\ell(u) + (p(r-1)+p(r))\ell(r-1)\right] + p(r-1) + p(r) \\ &= \mathbb{E}[\ell(V)] + p(r-1) + p(r) \end{aligned}$$

Note in the third step that for $r-1$ and r we add one bit to the code and for $1 \leq u \leq r-2$ the length function is the same. Thus, we have proven that $\mathbb{E}[\ell_{split}(u)] = \mathbb{E}[\ell(V)] + p(r-1) + p(r)$.

□

Observation: The Huffman code for U can be obtained from the Huffman code for V by "splitting".

The proof will be completed in the next lecture.